PENDING CLAIMS AS AMENDED

Please amend the claims as follows:

1. (currently amended) An apparatus having a KASUMI round circuit for generating a fractional portion of ~~the~~ a KASUMI cipher operably coupled to a calculation controller for sequencing ~~eight rounds~~ the KASUMI round circuit to produce a KASUMI output.

2. (currently amended) The apparatus of claim 1, further comprising a sub-key generator, connected to the calculation controller, for producing sub-keys ~~each round~~ for use by the KASUMI round circuit.

3. (currently amended) An apparatus for performing KASUMI ciphering on a KASUMI input with a key to produce a KASUMI output, comprising:

a KASUMI round circuit for generating a fractional portion of ~~the~~ a KASUMI cipher, configurable for calculation of even and odd rounds~~, eight rounds being calculated~~ to produce the KASUMI output;

memory for storing the output of the KASUMI round circuit; and

a selector for providing the input to the KASUMI round circuit, the KASUMI input being selected during the first round and ~~the contents of~~ a stored KASUMI round circuit output from the memory being selected during subsequent rounds.

4. (currently amended) The apparatus of claim 3, further comprising a sub-key generator for generating sub-keys for ~~each round~~ the KASUMI round circuit based on the key.

5. (original) The apparatus of claim 3, wherein the apparatus is adapted to be operable in an access point.

6. (original) The apparatus of claim 3, wherein the apparatus is adapted to be operable in an access terminal.

7. (original) The apparatus of claim 3, wherein the apparatus is adapted to be operable in a W-CDMA system.

8. (currently amended) A KASUMI round circuit for receiving an input and producing an output, operable with a partial round calculator from which the output is produced, comprising:

a memory for storing an intermediate value from the partial round calculator; and

a selector for selecting between the input and ~~the contents of~~ a stored intermediate value from the memory for delivery to the partial round calculator.

9. (currently amended) A KASUMI round circuit for receiving a 64-bit input and producing a 64-bit output comprising:

an FO ~~function~~ circuit having an FO output;

an FL ~~function~~ circuit having an FL output;

an XOR gate having a first XOR operand, a second XOR operand, and an XOR output;

a first register having a first register output;

a second register, having a second register output, the second register ~~for~~ receiving ~~the output of~~ the XOR ~~gate~~ output, the second register output being concatenated with the first register output ~~of the first register~~ to produce the 64-bit output;

a first input mux having a first input mux output, the first input mux ~~for~~ selecting between an ~~the~~ upper half of the 64-bit input and the ~~output of the~~ second register output, under control of an input select signal, the first input mux output being received at the first register;

a second input mux having a second input mux output, the second input mux ~~for~~ selecting between a ~~the~~ lower half of the 64-bit input and the output of the first register under control of the input select signal, the second input mux output being delivered as the second XOR operand ~~to the XOR gate~~;

a first datapath mux having a first datapath mux output, the first datapath mux~~, the output of which is delivered to the FL function, for~~ selecting between the ~~output of the~~ first input mux output and the ~~output of the~~ FO ~~function~~ output under control of a data flow signal, the first datapath mux output delivered to the FL circuit;

a second datapath mux having a second datapath mux output, the second datapath mux~~, the output of which is delivered to the FO function, for~~ selecting between the ~~output of the~~ FL

output ~~function~~ and the ~~output of the~~ first register output under control of the data flow signal, the second datapath mux output delivered to the FO circuit; and

a third datapath mux having a third datapath mux output, the third datapath mux, ~~the output of which is delivered as the first operand to the XOR gate,~~ for selecting between the ~~output of the~~ FL ~~function~~ output and the FO ~~function~~ output under control of the data flow signal.

10. (currently amended) An ~~FO function~~ apparatus for receiving an input and producing an output, operable with a partial FO calculator from which the output is produced, comprising:

a memory for storing an intermediate value from the partial FO calculator; and

a selector for selecting between the input and ~~the contents of~~ a stored intermediate value from the memory for delivery to the partial FO calculator.

11. (currently amended) An ~~FO function~~ apparatus for receiving a 32-bit input and producing a 32-bit output, comprising:

a first XOR gate having a first operand, a second operand, and a first XOR output, the first operand of the first XOR gate ~~for~~ receiving a KO sub-key ~~as a first operand~~;

an FI ~~function~~ circuit having an FI input and an FI output, the FI input ~~for~~ receiving the ~~output of the~~ first XOR ~~gate~~output;

a second XOR gate having a first operand, a second operand, and a second XOR output, the first operand of the second XOR gate ~~for~~ receiving the ~~output of the~~ FI ~~function~~ output ~~as a first operand~~;

a first register having a first register output, the first register ~~for~~ receiving the ~~output of the~~ second XOR ~~gate~~ output;

a second register having a second register input and a second register output;

a first input mux having a first input mux output, the first input mux, ~~the output of which is delivered as a second operand to the first XOR gate, for~~ selecting between ~~the~~ an upper half of the 32-bit input and the ~~output of the~~ second register output under control of an input select signal, the first input mux output delivered to the second operand of the first XOR gate; and

a second input mux having a second input mux output, the second input mux, ~~the output of which is delivered as the second operand to the second XOR gate and the input to the second register, for~~ selecting between ~~the~~ a lower half of the 32-bit input and the ~~output of the~~ first

register output under control of the input select signal, the second input mux output delivered as the second operand of the second XOR gate and delivered as the second register input, the second input mux output ~~being~~ concatenated with the ~~output of the~~ second XOR output ~~gate~~ to produce the 32-bit output.

12. (currently amended) An ~~FI function~~ apparatus for receiving an input and producing an output, operable with a partial FI calculator from which an intermediate value and the output is produced, comprising:

a memory for storing the intermediate value from the partial FI calculator; and

a selector for selecting between the input and ~~the contents of~~ a stored intermediate value from the memory for delivery to the partial FI calculator.

13. (currently amended) An ~~FI function~~ apparatus for receiving a 16-bit input and producing a 16-bit output, comprising:

a first register having a first register input and a first register output;

a second register having a second register input and a second register output;

a first input mux having a first input mux output, the first input mux ~~for~~ selecting between the ~~output of the~~ first register output and ~~the~~ an upper nine bits of the 16-bit input under control of an input select signal;

a second input mux having a second input mux output, the second input mux ~~for~~ selecting between the ~~output of the~~ second register output and ~~the~~ a lower seven bits of the 16-bit input under control of the input select signal;

an S9 ~~function~~ circuit, having an S9 output, ~~for~~ receiving the ~~output of the~~ first input mux output;

a first XOR having a first operand, a second operand, and a first XOR output, ~~for receiving as operands~~ the first operand of the first XOR receiving the ~~output of the~~ S9 ~~function~~ output and the second operand of the first XOR receiving the zero-extended ~~output of the~~ second input mux output;

an S7 ~~function~~ circuit, having an S7 output, for receiving the ~~output of the~~ second input mux output;

a second XOR having a first operand, a second operand, and a second XOR output, ~~for receiving as operands~~ the first operand of the second XOR receiving the truncated ~~output of the~~ first XOR output and the second operand of the second XOR receiving ~~the output of~~ the S7 ~~function~~ output, the second XOR output ~~being~~ concatenated with the ~~output of the~~ first XOR output to produce the 16-bit output;

a third XOR having a first operand, a second operand, and a third XOR output, the third XOR output ~~of which is~~ delivered to the first register input, ~~for receiving as operands~~ the first operand of the third XOR receiving a first KI sub-key, and the second operand of the third XOR receiving ~~and the output of~~ the first XOR output; and

a fourth XOR having a first operand, a second operand, and a fourth XOR output, the fourth XOR output ~~of which is~~ delivered to the second register input, ~~for receiving as operands~~ the first operand of the fourth XOR receiving a second KI sub-key, and the second operand of the fourth XOR receiving ~~and the output of~~ the second XOR output.

14. (currently amended) An ~~sub-key generator~~ apparatus for receiving a key and~~, for each round,~~ producing eight sub-keys for delivery to a KASUMI round circuit, comprising:

a first shift register having a first shift register output, loadable with the key, left rotatable by eight bits under control of a rotate signal, ~~four~~ a first subset of the eight sub-keys being derived from the first shift register output; and

a second shift register having a second shift register output, loadable with a masked version of the key, left rotatable by eight bits under control of the rotate signal, ~~four~~ a second subset of the eight sub-keys being derived from the second shift register output.

15. (original) A sub-key generator for receiving a 128-bit key and, for each round, producing eight 16-bit sub-keys, comprising:

a first 128-bit shift register, loadable with the key;

a second 128-bit shift register, loadable with a masked version of the key, the first sub-key being produced from the third highest 16 bits, the second sub-key being produced from the fourth highest 16 bits, the third sub-key being produced from the fifth highest 16 bits, the fourth sub-key being produced from the lowest 16 bits;

a rotator for rotating left by one bit the contents of the first shift register located in highest 16 bits to produce the fifth sub-key; a rotator for rotating left by five bits the contents of the first shift register located in the second highest 16 bits to produce the sixth sub-key;

a rotator for rotating left by eight bits the contents of the first shift register located in the third lowest 16 bits to produce the seventh sub-key; and

a rotator for rotating left by 13 bits the contents of the first shift register located in the second lowest 16 bits to produce the eighth sub-key.

16. (currently amended) A method for performing KASUMI ciphering on a KASUMI input, having an upper half and a lower half, to produce a KASUMI output, comprising:

for each of eight rounds:

selecting the KASUMI input ~~for use in the calculating step~~ as a calculation input during the first round;

selecting ~~the~~ a stored result ~~for use in the calculating step~~ as the calculation input in subsequent rounds;

calculating a partial result in a KASUMI round circuit with the selected calculation input; and

storing the partial result as the stored result in a memory; and

delivering the stored result as the KASUMI output.

17. (currently amended) The method of claim 16, wherein the calculating step comprises:

when the round is odd:

performing ~~the~~ an FL function on ~~the~~ an upper half of the ~~input or stored result, as selected in the selecting step~~ selected calculation input to produce an FL output;

performing ~~the~~ an FO function on the ~~output of the~~ FL ~~function~~ output to produce an FO output; and

XORing the ~~output of the~~ FO ~~function~~ output with ~~the~~ a lower half of the ~~input or stored result, as selected in the selecting step~~ selected calculation input to produce an XOR output;

when the round is even:

performing the FO function on ~~the~~ an upper half of the stored result to produce the FO output;

performing the FL function on the ~~output of the~~ FO ~~function~~output to produce the FL output; and

XORing the ~~output of the~~ FL ~~function~~ output with ~~the~~ a lower half of the stored result to produce the XOR output;

delivering as the partial result the ~~output of the XORing step~~ XOR output concatenated with the upper half of the ~~input or stored result, as selected in the selecting step~~ selected calculation input.

18. (original) The method of claim 16, further comprising generating sub-keys for each round.

19. (currently amended) A method for performing ~~the~~ an FO function comprising:

for each of three stages:

selecting ~~the~~ an input ~~for use in the calculating step~~ as a calculation input in the first stage;

selecting ~~the~~ a stored result ~~for use in the calculating step~~ as the calculation input in subsequent stages;

calculating a partial result with the selected calculation input;

storing the partial result as the stored result in memory; and

delivering the partial result as the output~~;~~ .

20. (currently amended) The method of claim 19, wherein the calculating step comprises:

XORing ~~the~~ an upper half of the ~~input or stored result, as selected in the selecting step,~~ selected calculation input with a sub-key to form a first XOR result;

performing ~~the~~ an FI function on the ~~output of the XORing step~~ first XOR result to form an FI result;

XORing the ~~output of the FI function~~ FI result with ~~the~~ a lower half of the ~~input or stored result, as selected in the selecting step~~ selected calculation input to form a second XOR result; and

delivering as the partial result ~~the~~ an upper half of the ~~input or stored result, as selected in the selecting step,~~ selected calculation input concatenated with the ~~output of the second XORing step~~ second XOR result.

21. (currently amended) A method for performing ~~the~~ an FI function comprising:

calculating a first partial result with a partial result circuit using ~~the~~ an input;

XORing the first partial result with a sub-key to form an XORed partial result;

storing the XORed partial result in a memory as a stored result;

calculating a second partial result with the partial result circuit using the stored result; and

delivering the second partial result as the output~~;~~ .

22. (currently amended) The method of claim 21, wherein the ~~calculating step comprises~~ partial result circuit is operable to perform the following:

performing ~~the~~ an S9 function on ~~the~~ an upper nine bits of the input or stored result;

zero extending ~~the~~ a lower 7 bits of the input or stored result;

XORing the zero-extended input or stored result with the output of the S9 function to form a first XOR result;

perfoming the S7 function on the lower 7 bits of the input or stored result;

truncating the ~~output of the XORing step~~first XOR result;

XORing the truncated first XOR ~~output~~ result with the output of the S7 function to form a second XOR result; and

delivering as the partial result the ~~output of the second XORing step~~ second XOR result concatenated with the ~~output of the first XORing step~~ first XOR result.

23. (original) A method for generating sub-keys comprising:

loading two sub-key shift registers;

for each of eight rounds:

deriving the sub-keys from the sub-key shift registers; and

rotating the two sub-key shift registers left by eight bits.

24. (original) The method of claim 23, wherein the loading step comprises:

loading a key into the first sub-key shift register;

masking the key; and

loading the masked key into the second sub-key shift register.

25. (original) The method of claim 24, wherein the deriving step comprises:

deriving the first sub-key from the third highest set of 16 bits of the second shift register;

deriving the second sub-key from the fourth highest set of 16 bits of the second shift register;

deriving the third sub-key from the fifth highest set of 16 bits of the second shift register;

deriving the fourth sub-key from the lowest set of 16 bits of the second shift register;

rotating the highest set of 16 bits of the first shift register by one bit to the left to produce the fifth sub-key;

rotating the second highest set of 16 bits of the first shift register by five bits to the left to produce the sixth sub-key;

rotating the third lowest set of 16 bits of the first shift register by eight bits to the left to produce the seventh sub-key; and

rotating the second lowest set of 16 bits of the first shift register by thirteen bits to the left to produce the eighth sub-key.